A GRID GENERATION SYSTEM FOR MULTI-DISCIPLINARY DESIGN OPTIMIZATION

William T. Jones*
Jamshid Samareh-Abolhassani†
Computer Sciences Corporation
Hampton, Virginia 23666

Abstract

A general multi-block three-dimensional volume grid generator is presented which is suitable for Multi-Disciplinary Design Optimization. The code is fast, robust, highly automated, and written in ANSI "C" for platform independence. Algebraic techniques are used to generate and/or modify block face and volume grids to reflect geometric changes resulting from design optimization. Volume grids are generated/modified in a batch environment and controlled via an ASCII user input deck. This allows the code to be incorporated directly into the design loop. Generated volume grids are presented for a High Speed Civil Transport (HSCT) Wing/Body geometry as well a complex HSCT configuration including horizontal and vertical tails, engine nacelles and pylons, and canard surfaces.

Nomenclature

a,b = Blending functions
C = Wing Chord
c,d = Blending functions
e,f = Edge arclengths
g,h = Normalized edge arclengths
P = Interpolant normalization
$\overline{R}$ = Physical space coordinate vector
$\overline{r}$ = Grid point physical coordinate vector
s,t = Local patch parametric coordinates
U,V = Parametric space coordinates
u,v = Grid point parametric coordinates
$\overline{W}$ = Parametric space coordinate vector
$\overline{w}$ = Grid point parametric coordinate vector
X,Y,Z = Physical space coordinates
x,y,z = Physical coordinates of a grid point
λ = Distance squared
Ω = Gradient of λ
ξ,η = Interpolants

Subscripts

i,j = denotes position in computational space
l,m = denotes two computational directions of a face
$l1,m1$ = denote the minimum face indices in the given direction
$l2,m2$ = denote the maximum face indices in the given direction
r = denotes wing root

* Member of the Technical Staff, Member AIAA
† Computer Scientist, Member AIAA

Introduction

The continuing increase in computer speeds and the advancement of numerical algorithms has helped to increase the interest in Multi-disciplinary Design Optimization (MDO). MDO is a methodology for the design of complex engineering systems and subsystems that coherently exploits the synergism of mutually interacting phenomena. The process involves analyzing independent solutions generated after perturbing the design variables of a baseline system.

Typically, the MDO process proceeds in an iterative manner. Each cycle, at a minimum, includes the generation of numerical solutions, determination of design sensitivities, and system optimization. It is apparent that a large number of cycles may be necessary to complete the optimization, and as a result, the process must be capable of efficient execution in the batch environment, with as little human intervention as possible.

Perturbations are based on design sensitivities which are derivatives of design parameters (e.g. lift) with respect to the design variables (e.g. airfoil camber). In practice, design sensitivities from the various disciplines used in the system analysis are obtained separately. These are then combined using the chain rule to obtain the global sensitivity of the

system. The derivatives may be obtained analytically, by differentiating the analysis code(s), or numerically, using finite differences. When using finite differences, however, it is often difficult to determine the appropriate step size to be used for a given design variable.

In many applications of MDO, computational fluid dynamics (CFD) is an integral part of the design process. As a result, it is necessary to possess a rapid and highly automated grid generation capability which produces changes in the surface and volumetric grids to reflect the perturbations of the baseline system. It is also desirable to make grid modifications within the design cycle. Many design parameters, such as lift and drag, are fairly well established after a small number of flow iterations. With geometric changes occurring within the design loop, partially converged flow solutions can be used as a starting solution for the next cycle, thereby reducing the overall computational effort[1].

Current grid generation techniques have been strongly shaped by a push to develop interactive tools which aid in the discretization of computational domains. While most of these tools are well suited for the generation of grids about unique configurations, their generality requires a large degree of human interaction. Even for the smallest changes, like those resulting between MDO cycles, a great deal of input of varying degree is needed to redefine the computational structure. Recently, some tools[2,3] have incorporated useful parametric capabilities, but continue to rely on interactive software control. These methods fall short of satisfying the rapid, "hands-off" grid generation needs of CFD in the MDO cycle.

In this paper, the development of a computer program specifically designed for MDO grid generation is described. The code is capable of generating CFD quality surface and volume grids, as well as analytic grid sensitivities with respect to the design variables. In addition, existing grids generated with other grid generation systems may be modified to reflect the geometric changes defined in an automated design and optimization cycle. This program has been applied in the generation and modification of both inviscid and viscous CFD grids.

In the sections to follow, the approach taken to develop the Coordinate and Sensitivity Calculator for Multi-disciplinary Design Optimization (CSCMDO) computer program is described. Also discussed are results from aerospace test cases involving High Speed Civil Transport (HSCT) configurations and future directions of code development.

Approach

The CSCMDO code includes specialized features required by MDO grid modifications. These features are added to compliment standard volume grid generation methods. The code is capable of modifying any of the six faces of a block, representing a computational cube, to reflect geometric changes in the optimized system as defined by input surface(s). It is controlled via an ASCII user input file for execution in a batch environment.

Figure 1 shows the layout of a typical MDO design loop. Information input from outside the loop is generated one time before the loop is initiated. This information includes the baseline geometry, baseline CFD grid, and a user input file. The design loop is totally self-contained and therefore requires no human intervention. The CSCMDO code operates within the loop to provide automated volume grid generation / modification within the each design cycle.

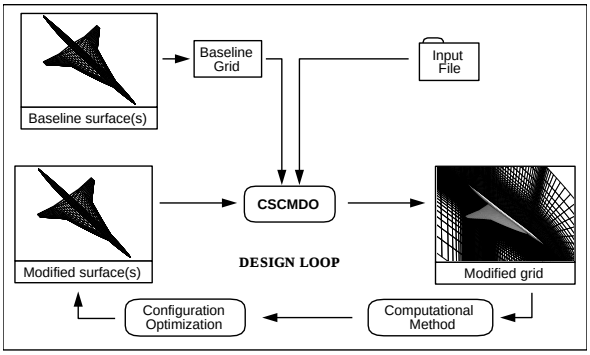


Figure 1 - Integration of grid generation into the design loop

The baseline surfaces are provided in the form of a structured mesh of discrete point data. The number and distribution of points defining the surfaces are not required to match those of the desired CFD grid. However, sufficient point resolution must be provided so as to adequately define all surface curvature. Some methods of surface modification described herein will require that the surface mesh topology match that of the desired CFD grid. Also, points of interest, such as a wing crank location, must fall along isoparametric lines in the surface definition. Point continuous surface/surface intersections are desirable. In the event that point continuity cannot be provided at surface/surface intersections, intersection curves on each surface should have sufficient resolution such that interpolation errors resulting from the inconsistent data do not contaminate the quality of the resulting CFD grid.

The baseline grid may be in the form of a baseline volume grid or, in some instances, discrete grid point data for the six faces of each block. The original grids can be generated using any structured grid generation package. Available file formats

include some of the most widely used CFD software.

Within the design loop, the optimization process is to provide the modified surface geometry definition, again as discrete point data[4-7]. The changes represented by the modified surfaces are assumed to be small so as not to violate the original topology definition. In the event the modifications do violate the topology, grid quality checks provide return codes to the software controlling the loop for appropriate action. Standard algebraic two-dimensional(2D) and three-dimensional(3D) grid generation functionality is implemented for the generation of faces and blocks. Blocks may be automatically subdivided using intermediate face "hard planes" to increase the quality of 3D interpolation. Hard planes may be input or calculated internally.

Face Modifications

Face modifications are provided using a variety of methods to be described in the following sections. These methods include simple input of a modified face, parametric updates to a modified surface, projection to a modified surface, and deformations conforming to a modified surface.

Parametric - The parametric updates[8] include methods utilizing mappings to uniform parameter space (UPS), as well as arclength parameter space (APS). APS is provided in the event that input surface parameterization is not smooth and orthogonal. The use of UPS with such a surface could produce an unacceptable CFD grid. A detailed explanation is contained in reference 8. APS seeks to alleviate such shortcomings.

These methods use a forward mapping into the selected parameter space where the grid generation takes place. Parameter values are then mapped back to physical space according to the modified surface.

The mapping to and from both UPS and APS, as well as a description of the parameter space grid generation technique of reference 8, are briefly described here for completeness.

Forward mapping involves the transfer of the surface grid from physical space $\left(\overline{R} = \{X, Y, Z\}^T\right)$ to a parameter space $\left(\overline{W} = \{U, V\}^T\right)$. The surface is a parametric surface such as a bilinear, bicubic, or NURBS representation. UPS is the primary type of parameter space used in grid generation, and is simply constructed using the surface grid indices according to

$$U_{i,j} = i,, \qquad V_{i,j} = j. \qquad (1)$$

APS is constructed using the surface arclengths in each computational direction as

$$\begin{aligned} U_{i,j} &= U_{i-1,j} + e_{i,j}, \\ V_{i,j} &= V_{i-1,j} + f_{i,j}, \end{aligned} \qquad (2)$$

where

$$\begin{aligned} e_{i,j} &= \left\| \overline{R}_{i,j} - \overline{R}_{i-1,j} \right\|, \\ f_{i,j} &= \left\| \overline{R}_{i,j} - \overline{R}_{i,j-1} \right\|. \end{aligned} \qquad (3)$$

Grid generation is the same for both parameter spaces. The edges of the CFD face are defined in the appropriate parameter space. In CSCMDO, the orientation of the CFD face to the background surface is automatically determined. A manual override is provided in the event the orientation search fails. Once the edges are known, the interior of the face is generated using arclength based transfinite interpolation (TFI). With TFI, the interior grid points are defined by the edge and corner contributions according to

$$\begin{aligned} \overline{w}_{l,m} = {} & a_{l,m}\overline{w}_{l,m1} + b_{l,m}\overline{w}_{l,m2} + c_{l,m}\overline{w}_{l1,m} + d_{l,m}\overline{w}_{l2,m} \\ & - a_{l,m}c_{l,m}\overline{w}_{l1,m1} - b_{l,m}c_{l,m}\overline{w}_{l1,m2} \\ & - a_{l,m}d_{l,m}\overline{w}_{l2,m1} - b_{l,m}d_{l,m}\overline{w}_{l2,m2}. \end{aligned} \qquad (4)$$

The subscripts ($l1$,$l2$) and ($m1$,$m2$) are the minimum and maximum indices of the face in the u and v directions respectively. The variables a,b,c and d are the blending functions subject to the following conditions,

$$\begin{aligned} a_{l,m1} = c_{l1,m} = b_{l,m2} = d_{l2,m} = 1, \\ a_{l,m2} = c_{l2,m} = b_{l,m1} = d_{l1,m} = 0. \end{aligned} \qquad (5)$$

A very robust set of blending functions has been proposed by Soni[9]. These blending functions are based on the normalized arclength of the edges in physical space, and are defined as

$$\begin{aligned} a_{l,m} &= 1 - \eta_{l,m}, \qquad b_{l,m} = \eta_{l,m}, \\ c_{l,m} &= 1 - \xi_{l,m}, \qquad d_{l,m} = \xi_{l,m}, \end{aligned} \qquad (6)$$

where

$$\begin{aligned} \xi_{l,m} &= \frac{g_{l,m1} + h_{l1,m}\left\{g_{l,m2} - g_{l,m1}\right\}}{P_{l,m}}, \\ \eta_{l,m} &= \frac{h_{l1,m} + g_{l,m1}\left\{h_{l2,m} - h_{l1,m}\right\}}{P_{l,m}}, \\ P_{l,m} &= 1 - \left\{g_{l,m2} - g_{l,m1}\right\}\left\{h_{l2,m} - h_{l1,m}\right\}, \end{aligned} \qquad (7)$$

The values of $g_{l,m}$ and $h_{l,m}$ are the normalized edge arclengths in physical space. With this interpolation complete, the grid can be mapped back from the parameter space to the physical space.

Backward mapping is used to transfer the grid points from the parametric space $\left(\bar{w}_{l,m} = \{u_{l,m}, v_{l,m}\}^T\right)$ to the physical space $\left(\bar{r}_{l,m} = \{x_{l,m}, y_{l,m}, z_{l,m}\}^T\right)$. The backward mapping involves three steps. The first step is a search of the surface parameter space to determine the local surface patch which includes the current grid point. The second step is to compute the local patch coordinates $(s_{l,m}, t_{l,m})$ for the coordinates in parametric space $(\bar{w}_{l,m})$. For UPS this is relatively simple, where

$$\begin{aligned} s_{l,m} &= u_{l,m} - U_{i,j}, \\ t_{l,m} &= v_{l,m} - V_{i,j}. \end{aligned} \tag{8}$$

For APS, the local coordinates are computed using a set of quasi-linear equations. The UPS parameter values of $U_{i,j}$ and $V_{i,j}$ are known for each corner of the patch and the patch grid point $(u_{l,m}, v_{l,m})$ is known in the interior of the patch. With this information a system of two equations with two unknowns is formed, where

$$\begin{aligned} u_{l,m} &= (1-s_{l,m})(1-t_{l,m})U_{i,j} + s_{l,m}(1-t_{l,m})U_{i+1,j} \\ &\quad +(1-s_{l,m})t_{l,m}U_{i,j+1} + s_{l,m}t_{l,m}U_{i+1,j+1}, \\ v_{l,m} &= (1-s_{l,m})(1-t_{l,m})V_{i,j} + s_{l,m}(1-t_{l,m})V_{i+1,j} \\ &\quad +(1-s_{l,m})t_{l,m}V_{i,j+1} + s_{l,m}t_{l,m}V_{i+1,j+1}. \end{aligned} \tag{9}$$

The solution to these equations is obtained using the Newton-Raphson method and gives the local coordinates of the patch $(s_{l,m}, t_{l,m})$. The final step is to compute the physical coordinates $(\bar{r}_{l,m})$ of the grid point on the local surface patch using bilinear or bicubic interpolation.

Projection - Projection of a face allows for the generation of a CFD grid over several surfaces, while also avoiding problems associated with surface parameterization mentioned in the previous section. The projection algorithm[10] used in CSCMDO is described briefly for completeness. Again the surface is approximated by a parametric surface as,

$$\begin{aligned} \bar{R}(\bar{w}) &= \{x(\bar{w}), y(\bar{w}), z(\bar{w})\}^T, \\ \bar{w} &= \{u, v\}^T. \end{aligned} \tag{10}$$

The process of projecting a point, $\bar{r}$, onto a surface $\bar{R}(\bar{w})$, involves finding $\bar{w}$ such that the distance, d, between $\bar{r}$ and $\bar{R}(\bar{w})$ is minimized given that it does not violate the limits of $\bar{w}$. The distance, d, can be written in terms of parameter $\bar{w}$ as,

$$d^2(\bar{w}) = \lambda(\bar{w}) = \left|\bar{R}(\bar{w}) - \bar{r}\right|^2. \tag{11}$$

To minimize d, Eq. (11) must be minimized with respect to $\bar{w}$. This is accomplished by setting the gradient of λ, $\nabla f(\bar{w})$, equal to zero, as

$$\begin{aligned} \nabla f(\bar{w}) = \Omega_i(\bar{w}) &= \frac{\partial f(\bar{w})}{\partial u_i} = 0, \\ \Omega_i &= \frac{\partial \bar{R}(\bar{w})}{\partial u_i} \cdot \left\{\bar{R}(\bar{w}) - \bar{r}\right\}. \end{aligned} \tag{12}$$

Solution of the above nonlinear system of equations for $\bar{w}$ results in the projected point.

As with the parametric update, a surface is represented as a set of bilinear or bicubic patches. For each patch, $\bar{R}(u,v)$ is approximated in terms of its parameters as,

$$\begin{aligned} \bar{R}(u,v) &= (1-u)(1-v)\bar{R}_{i,j} + u(1-v)\bar{R}_{i+1,j} \\ &\quad +(1-u)v\bar{R}_{i,j+1} + uv\bar{R}_{i+1,j+1}. \end{aligned} \tag{13}$$

Combining Eqs. (12) and (13) yields the following system of nonlinear equations,

$$\frac{\partial \bar{R}}{\partial u} \cdot \left\{\bar{R}(\bar{w}) - \bar{r}\right\} = 0, \qquad \frac{\partial \bar{R}}{\partial v} \cdot \left\{\bar{R}(\bar{w}) - \bar{r}\right\} = 0, \tag{14}$$

where,

$$\begin{aligned} \frac{\partial \bar{R}}{\partial u} &= (1-v)\left(\bar{R}_{i+1,j} - \bar{R}_{i,j}\right) + v\left(\bar{R}_{i+1,j+1} - \bar{R}_{i,j+1}\right), \\ \frac{\partial \bar{R}}{\partial v} &= (1-u)\left(\bar{R}_{i,j+1} - \bar{R}_{i,j}\right) + u\left(\bar{R}_{i+1,j+1} - \bar{R}_{i+1,j}\right). \end{aligned} \tag{15}$$

Again the system is solved by the Newton-Raphson method. An additional method of projection available in CSCMDO is the projection onto an original surface(s) and evaluation based on a new surface(s). This method obtains the parametric coordinates by projecting each face grid point onto the baseline surface. The parametric coordinates are then mapped to the physical space using the surface(s) driving the modification. The resulting

grid maintains the same characteristics of the original grid. The shortcoming of this method, however, is the required existence of both the original and modified surfaces at the time of projection.

Deformation - Deformation of a face is key in providing reusability of an existing CFD grid. When a face is modified, a comparison to the original face edge points is made according to,

$$\Delta\overline{R} = \overline{R}_{\mathrm{mod}\,ified} - \overline{R}_{original}, \qquad (16)$$

Rearranging Eq. (16), the modified edge is given by,

$$\overline{R}_{\mathrm{mod}\,ified} = \overline{R}_{original} + \Delta\overline{R}. \qquad (17)$$

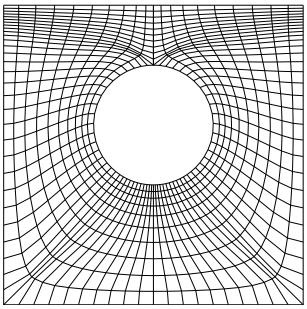
If all edges have been modified, Eqs. (16) and (17) are applied to the interior grid points to yield the modified face. In the event that less than 4 edges of a face are modified, unmodified edge $\Delta\overline{R}$ values in Eq. (17) are obtained by blending the ending values of $\Delta\overline{R}$ along the edge. The edge $\Delta\overline{R}$ values are then interpolated into the interior for use in Eq. (17). This method maintains the grid quality characteristics of the original CFD grid and provides time saving reusability of an optimal CFD grid.

Figure 2 shows the method applied to a C-H grid around a circle enclosed in a rectangular outer domain. The original grid in Figure 2-a shows smoothness and orthogonality characteristics obtained through partial differential equation (PDE) smoothing. Figures 2-b and 2-c show a modification to the circular inner edge. In Figure 2-b the edge is modified and the face is re-initialized with arclength based TFI. Figure 2-c shows the effect of a face deformation governed by the same change to the circular edge. Note the preservation of the smoothness and orthogonality characteristics in Figure 2-c. This method is extended to 3D as will be discussed in later sections.

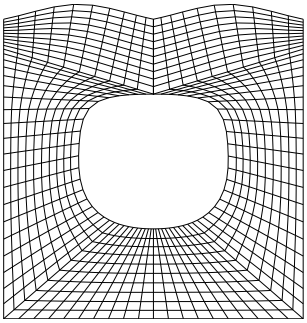
Block Modifications

Volume modifications are accomplished using standard algebraic methods for the re-initialization of the block interior, or by deformation of the original block interior based on changes defined by the six faces.

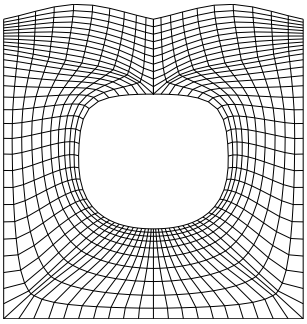
Grid Generation - The generation of the block interior is conducted in a manner which is an extension of the 2D methods of arclength based TFI described above. As previously mentioned, a block may be automatically subdivided by inserting intermediate faces, or "hard planes" within the block. Each hard plane acts to subdivide the block providing added control over the quality of developing interior. Hard planes may be directly input, as in the case where a PDE smoother was used to obtain an optimal grid, or generated internally using the 2D methods described above.



a) Original smoothed, orthogonal face



b) Modified face with algebraic TFI



c) Modified face with face deformations

Figure 2 - Example of Face deformation

Deformation - A method similar to that described for face deformation is extended to 3D for use in the deformation of entire blocks. Once the deformations are calculated for each of the six faces of a block, arclength based TFI is used to blend the deformations into the block interior. The deformations are then applied to the original grid points to obtain a deformed block. Again this method of deformation facilitates the reusability of an existing volume grid. The overall quality and characteristics of an existing block are maintained after face modification in the modified block.

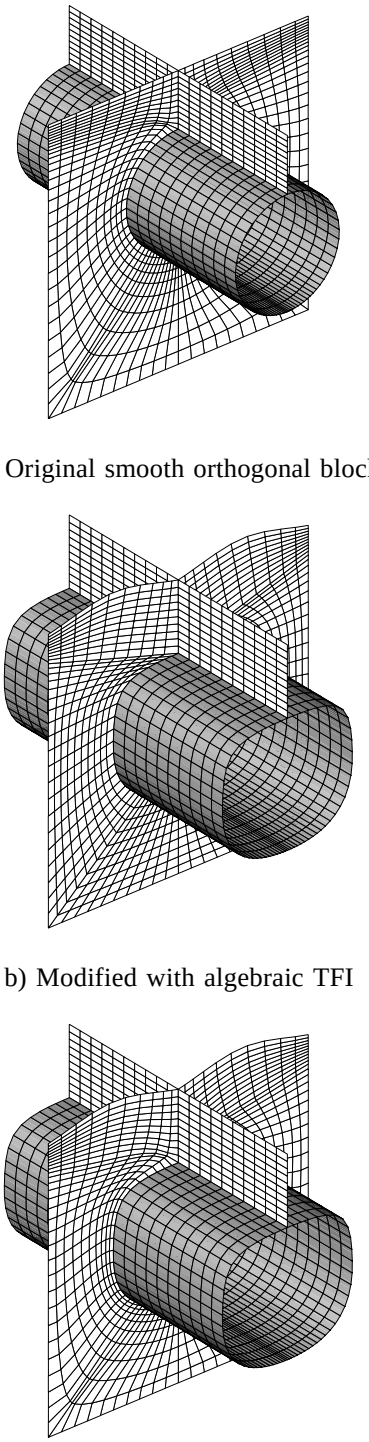
Figure 3 shows the 3D deformation applied to the same C-H topology used previously in 2D. Figure 3-a shows smoothness and orthogonality characteristics obtained through partial differential equation (PDE) smoothing for the original block. Figures 3-b and 3-c show the same modification to the cylindrical inner face. In Figure 3-b the face is modified with the adjacent faces and interior re-initialized with arclength based TFI. Figure 3-c shows the effect of face and block deformation governed by a change to the cylindrical inner face. Again note the preservation of the smoothness and orthogonality characteristics in Figure 3-c on the block interior.

Grid Quality

A check of grid quality measures is included in the code for diagnostic output as well as limited control when running in a batch environment. Cell volume and cell skewness are calculated on a block by block basis. A histogram of cell skewness is provided with the code output showing the percentage of the grid possessing a skewness over a range of 0^o to 90^o. Minimum cell volume, maximum cell skewness, and maximum average cell skewness controls are provided to trigger return codes which are used by the process controlling batch execution. The return codes allow for the controlling process to exit the design loop without wasting continued resources in the event the resulting volume grid is not suitable for use.

Grid Sensitivity

Calculation of grid sensitivity to the design variables involves the automatic differentiation of the CSCMDO code. This is accomplished using the Automatic DIfferentiation of C source, ADIC[11], processor from Argonne National Labs. ADIC is a tool for the automatic differentiation of ANSI C programs providing similar capabilities for C as ADIFOR[12-13] does for FORTRAN77.



a) Original smooth orthogonal block

b) Modified with algebraic TFI

c) Modified with deformation

Figure 3 Example of block deformation

The ADIC tool operates on a given source code using a specified set of dependent and independent variables, to produce an augmented C source that

computes not only the original scalar result, but also the partial derivatives of all of the specified dependent variables with respect to the independent variables. ADIC employs the Sage++ programming environment developed at Indiana University.

The cost of generating the sensitivities with the ADIC processed version of CSCMDO (CSCMDO.AD) has proven to be negligible when compared to the overall cost of a CFD design cycle. With an ADIC processed source code, the ratio between the cost of evaluating a gradient with n components and the cost of evaluating the original underlying scalar function is not n+1, but is bounded by 5 at most[14]. Preliminary testing of CSCMDO.AD has produced reasonable results. These results have been compared to finite difference calculations and show favorable agreement.

Viscous Grid Capabilities

The methods presented herein have been successfully applied to viscous grids. Care must be taken to provide surface geometry containing sufficient resolution to capture all surface curvature. The use of block deformation is very useful in the reuse of viscous grids which were originally refined using PDE techniques.

Results

CSCMDO has been extensively tested on aerospace configurations at the NASA Langley Research Center's (LaRC) Geometry Laboratory (GEOLAB). The test cases range from simple wing/body configurations to full HSCT geometry with tail surfaces, engine nacelles, and canards. The code is also used outside of the design loop in the GEOLAB for the rapid modification and quality check of existing CFD volume grids.

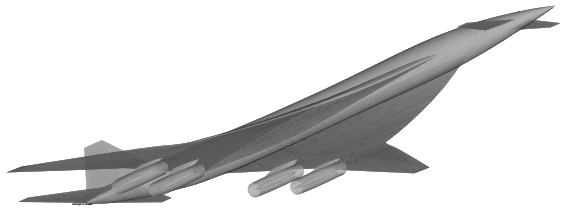


Figure 4 - Complex HSCT configuration

Execution time is difficult to determine for the code due to the wide variety of execution schedules available to the user. Typical timing, however, can be given as an example of two types of configurations.

The first is that of a simple double delta wing/body aircraft. The computational domain was broken into two C-H blocks with a total of 350,000 grid points. Seven modified geometries were supplied along with the baseline and the block topology. The geometry modifications included changes to wing sweep, camber, span, crank location, etc.. All eight volume grids, baseline and seven modifications, were generated sequentially on a Silicon Graphics Onyx workstation in double precision in under three minutes.

The second case is that shown in Figure 4. The configuration includes a double delta wing, tail surfaces, two flow through engine nacelles with a plug insert, engine pylons, and a canard. The computational domain contained seventeen blocks with at total of over 1.5 million grid points. This complex case required on the order of three minutes for the generation of a single complete volume grid with modifications to a variety of surfaces.

Figure 5 presents the results of the case involving the complex HSCT in Figure 4. Figure 5 shows the computed grid on the surface of the configuration. A cut from the 3D volume is shown in Figure 6. The view is looking downstream at the nose of the aircraft.

Figure 7 shows the computational grid used for grid sensitivity calculation testing. The grid is a two block H-O type around a wing/body/tails geometry generated with the RAPID[7] code. A sensitivity enhanced version of RAPID, from ADIFOR, was used to obtain both the surface definition and its sensitivity with respect to a change in wing root chord, C_r. Computed sensitivity contours are shown in Figure 8. Within RAPID, the trailing edge of the wing was fixed with changes propagating only forward. This is evident from the X sensitivity contours of Figure 8-a. Note the symmetry in Figures 8-a and 8-b. Only surface contours are shown for $\partial z / \partial C_r$ in Figure 8-c. This is due to the fact that a change in Cr has no effect on the X-Y symmetry plane shown in Figures 8-a and 8-b.

Future Directions

Future plans include the addition of PDE smoothing of block faces and entire blocks within the CSCMDO code. Also to be included are capabilities for complete domain decomposition within the code. Presently the block topology, number of points, and relative spacing are required input to CSCMDO. Future plans are to incorporate methods of defining

these key features of a computational domain within the program.

Conclusions

A structured CFD volume grid generator has been presented which is designed around the needs of multi-disciplinary design optimization. The multi-block code employs methods which make it ideal for the type of grid modifications required by MDO. Calculation of grid sensitivity data is highly accurate and efficient. The use of the ADIC processor to automatically differentiate the code greatly aids in the timely implementation of sensitivity calculations. The code has been used to generate/modify numerous CFD volume grids about a variety of configurations. The code is robust, fast, platform independent, and ready to meet the needs of multi-disciplinary design.

Acknowledgments

The authors would like to acknowledge the assistance provided by Christian Bischof and Andrew Mauer of Argonne National Laboratories. Their help in the area of automatic differentiation, and in the processing of CSCMDO with the ADIC tool, was crucial to the timely implementation of grid sensitivity calculations in CSCMDO.

References

1) Doria, Michael L., "An Investigation of Design Optimization using a 2-D Viscous Flow Code with Multigrid", NASA/ASEE Summer Faculty Fellowship Program, 1990.

2) Steinbrenner, John P., and Chawner, John R., "The GRIDGEN Version 9 Multiple Block Grid Generation Software", MDA Engineering Report 94-01, 1994.

3) Bertin, D., Casties, C., and Lordon, J., "A New Automatic Grid Generation Environment for CFD Applications", Fourth International Conference on Numerical Grid Generation in CFD and Related Fields, Swansea, Wales, 1994, pp. 391,402.

4) Barger, Raymond L., and Adams, Mary S., "Automatic Computation of Wing-Fuselage Intersection Lines and Fillet Inserts with Fixed-Area Constraint", NASA TM-4406, March 1993.

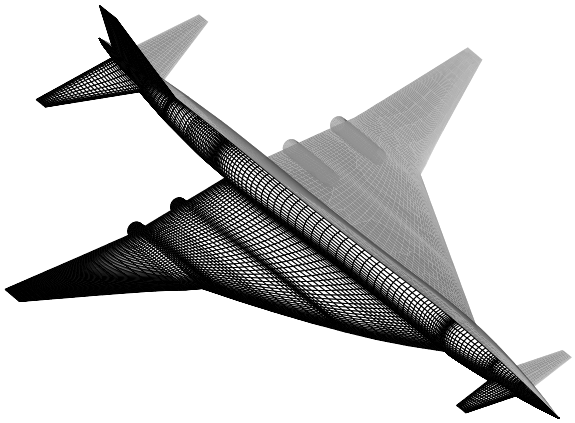


Figure 5 - CFD surface grid for complex geometry

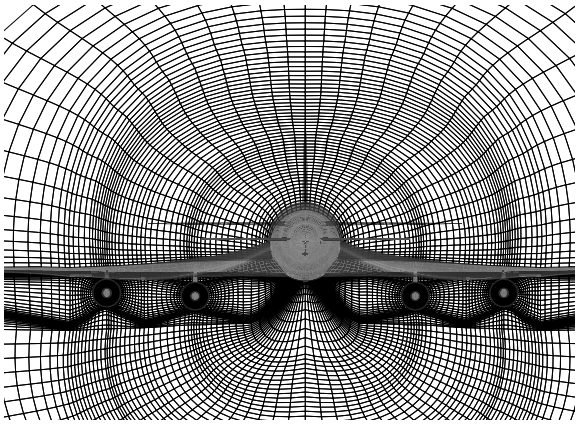


Figure 6 - Example of complex volume grid
(Grid from cut normal to flow)

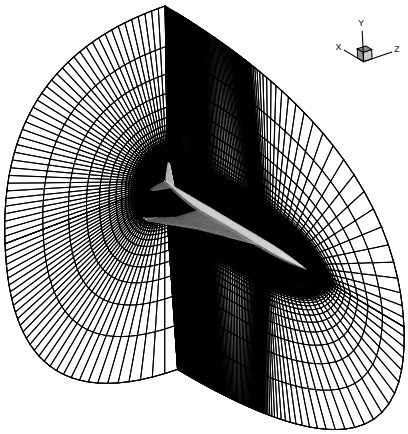


Figure 7 - Computational grid used in sensitivity test

5) Barger, Raymond L., and Adams, Mary S., "Automatic Computation of Euler-Marching and Subsonic Grids for Wing-Fuselage Configurations", NASA TM-4573, July 1994.

6) Barger, Raymond L., and Adams, Mary S., "Automatic Procedures for Computing Complete Configuration Geometry from Individual Component Descriptions", NASA TM-4607, July 1994.

7) Smith, Robert E., Bloor, Malcolm G. I., Wilson, Michael, and Thomas, Almuttil M., "Rapid Airplane Parametric Input Design (RAPID)", AIAA 95-1687, 1995

8) Samareh-Abolhassani, Jamshid, and Stewart, John E., "Surface Grid Generation in a Parameter Space", Journal of Computational Physics, Vol. 113, No. 1, July 1994.

9) Soni, B.K., "Two- and Three-Dimensional Grid Generation for Internal Flow Applications of Computational Fluid Dynamics", AIAA-85-1526, 1985.

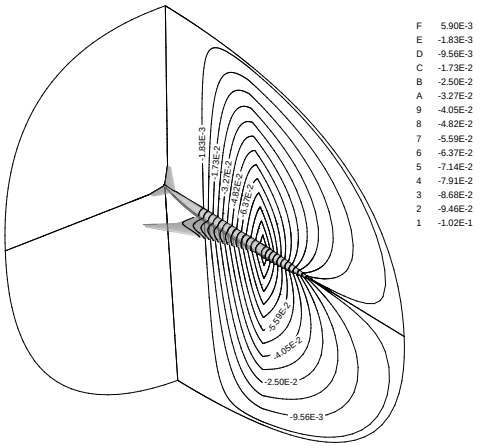
10) Samareh-Abolhassani, Jamshid, "Unstructured Grid on NURBS SURFACES", AIAA-93-3454, 1993.

11) Bischof, Christian, and Mauer, Andrew, "ADIC: A tool for the automatic differentiation of C Programs", Preprint MCS-P499-0295, Mathematics and Computer Science Division, Argonne National Laboratory, 1995.

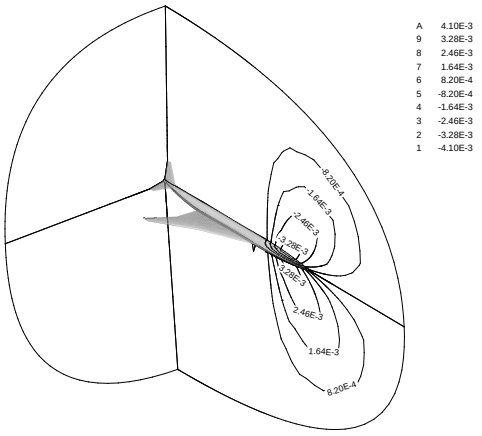
12) Bischof, Christian, Carle, Alan, Corliss, George, Griewank, Andreas, and Hovland, Paul, "ADIFOR: Generating Derivative Codes from FORTRAN Programs", Scientific Programming, Volume 1, Number 1, pp. 11,29, 1992.

13) Bischof, Christian, Carle, Alan, Khademi, Peyvand, and Mauer, Andrew. "The ADIFOR 2.0 system for the automatic differentiation of FORTRAN 77 Programs", Preprint MCS-P481-1194, Mathematics and Computer Science Division, Argonne National Laboratory, 1994.

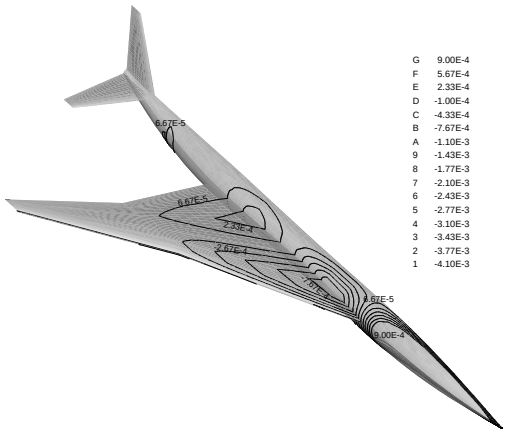
14) Griewank, Andreas, "On Automatic Differentiation", Mathematical Programming: Recent Developments and Applications, pages 83-108, Kluwer Academic Publishers, Amsterdam, 1989.



a) Field grid $\partial X / \partial Cr$ Sensitivity



b) Field grid $\partial Y / \partial Cr$ Sensitivity



c) Field grid $\partial Z / \partial Cr$ Sensitivity

Figure 8 - Field grid sensitivity with respect to wing root chord change